Card Pack Opener - Design Manual

Designed by Junior Weil, Joseph Helm, Brandon Cho, Brian Scotto

Introduction

The pack opening was implemented through JavaFX to create the visuals and was linked to the Java classes that were written to create functionality behind the visuals. The class CardSimController worked to code action events associated with the visuals. It worked to change the scene being viewed, draw cards when prompted, and navigate through the design. Many classes were implemented to control the opening of packs and assignment of variables. Each sport has a subclass of Card in order to read from the correct CSV files, and assign these variables to an ArrayList object. Then all of these subclasses were used in an overarching Pack class that included methods to draw the cards and assign variables to the corresponding attributes of each player. The Pack class was implemented into the CardSimController class and is what drives the functionality of the program. These classes interact with each other such as a real sports pack would, where any of the basketball cards could be drawn every time a basketball pack is opened.

User Stories

The user personas that would possibly be interested in our design are crucial to the development. Without knowing what would draw in potential users, there would be a lack of unique aspects to the program. There are a wide range of users between sports card collectors to curious individuals looking to possibly create their own collectible sports card. The main functionalities that stand out to users would be the ability to open packs of different sports cards

without having to spend money, as well as the ability to customize a card. The program does not solve practical problems, rather it is a way for users to pass time and simulate the thrill of opening sports card packs.

Object-Oriented Design

CRC Cards

Abstract **Card** SoccerCard, CustomCard, BasketballCard, FootballCard	
• Adding CustomCard classes to subclass Lists	• Sport, SoccerCard, FootballCard, BasketballCard, CustomCard

The card class interacts with every other class as it is the parent of all specific card classes. It also has a method to add a CustomCard to the pack opening possibilities.

Interface **Pack**	
• Draw cards to be displayed and assign their values to variables • add picture links to each of the cards to display	• Card, SoccerCard, FootballCard, BasketballCard, CustomCard

The pack class mainly works with the subclasses of Card and is the class that is implemented for functionality in the GUI. This class takes attributes from each of the subclasses and draws cards to be displayed.

CustomCard Card	
• Hold the variables given by the user to add a new card	• Card

The CustomCard class' only responsibility is to obtain the values necessary to be displayed on a card. This is added to the Pack through the Card class which puts it in the respective sport's card list.

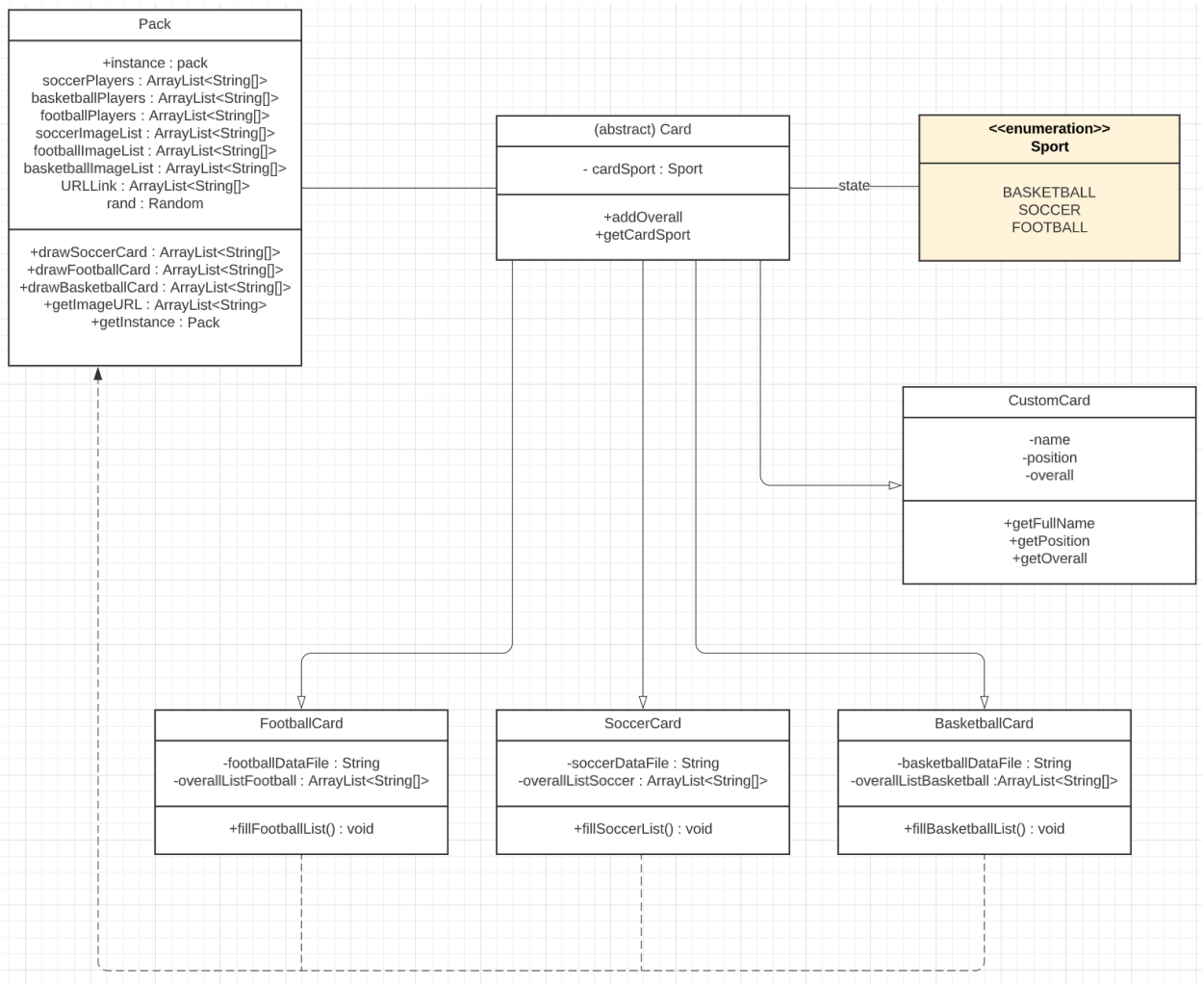
*SPORT*Card	Card
• Read in CSV file for respective sport • apply those values to a list • create a list of arrays with the attributes needed to display a card	• Card, Pack

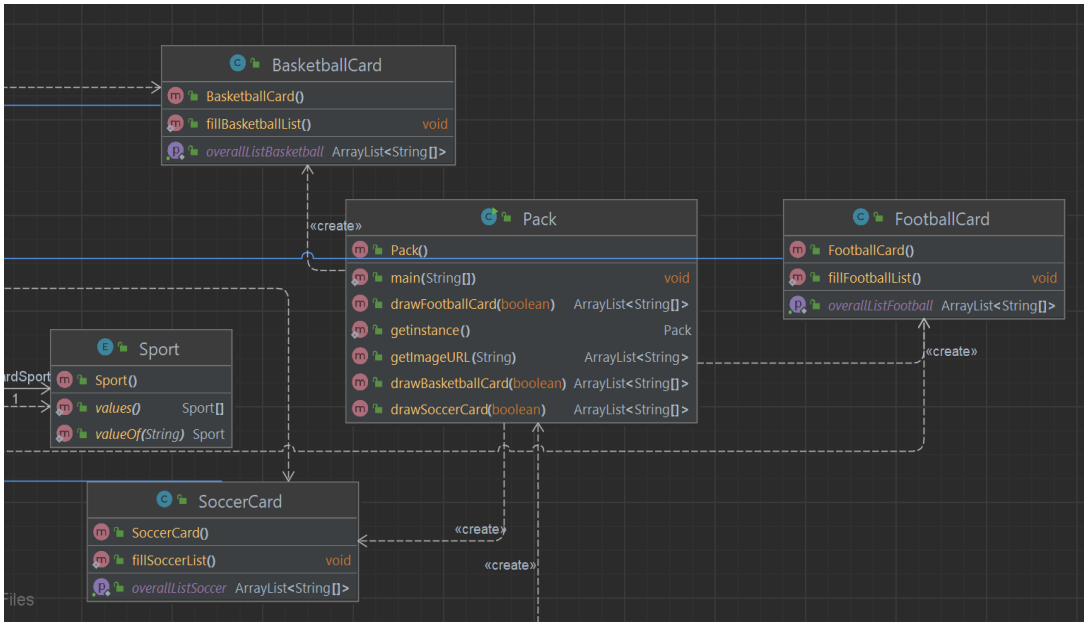
This class reads all of the CSVs for the respective sports and adds that data into an ArrayList of string arrays with the important values of each player. This class works with the Card class as the Card can add values and it inherits enumeration from the Card class. It works with the Pack class as each Pack instantiated creates a *SPORT*Card class for each sport in order to get possible values to draw from.

UML Diagrams

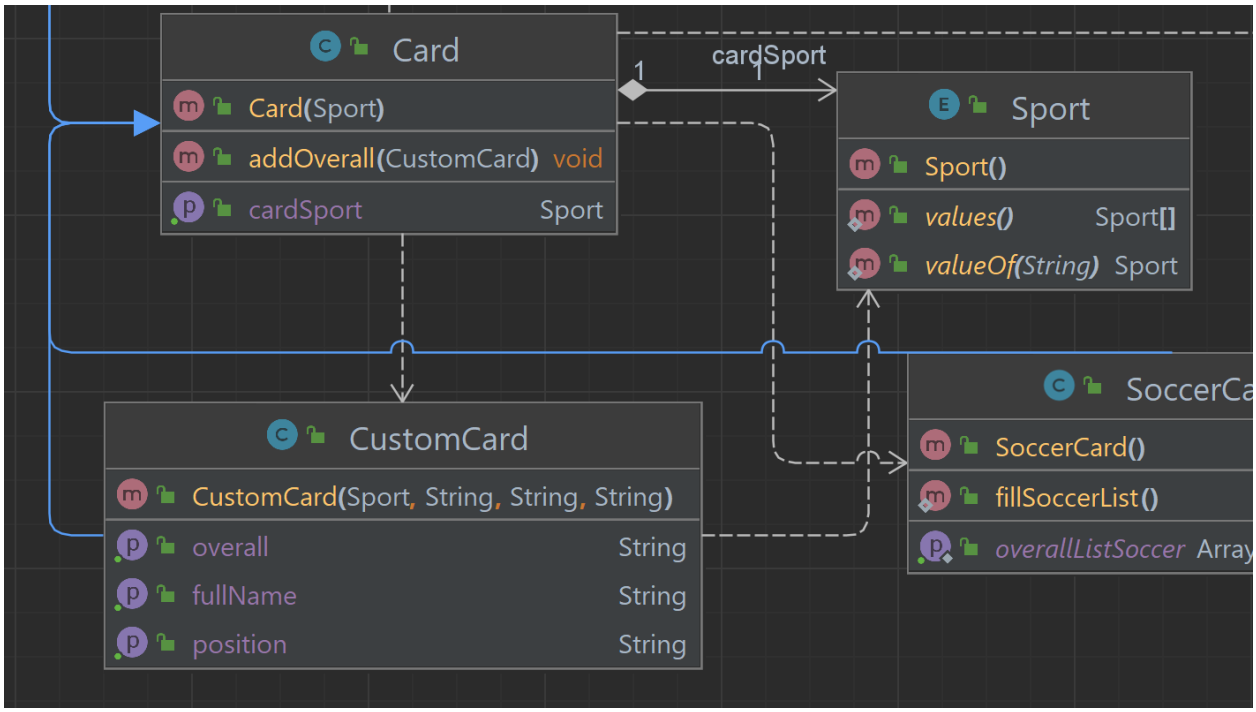
Below is the final UML diagram for the functional classes. Enumeration, inheritance, and dependence were all aspects of object oriented programming that were implemented through this multi-class structure. Enumeration was used in order to correctly identify the subclass of card to add the value to. The enumeration was to determine the sport of a custom card that could identify with any of the three subclasses of Card. The Pack class depends on each of the card subclasses as they are instantiated in the constructor of the class and used when drawing cards from the lists that are provided by methods in the subclasses. The *SPORT*Card classes inherit from Card and the main link between them is the enumeration that is inherited. The addOverall method depends

on the *SPORT*Card classes because it needs a certain class to add to its list. Below is the connection between these classes in UML form.





This screenshot from the auto-generated UML diagram from IntelliJ shows the dependencies of the Pack class. It shows that it is dependent on each of the card subclasses as they are used to create the list of cards to be drawn from. The arrow going into the Pack class is from CardSimController which depends on only the Pack class for implementation.



This screenshot depicts the classes behind the custom card and their dependencies. It also shows the importance of the enumeration and what classes are dependent on that. The custom card is a unique implementation that impacts user persona's desire to use the program.